09/360,542

	L #	Hits	Search Text	DBs	Time Stamp
1	L1	327341	recover$4	USPAT; US-PGPUB	2002/01/31 16:22
2	L2	55802	backup or back-up or (backing adj1 up)	USPAT; US-PGPUB	2002/01/31 16:22
3	L3	144108	restor$4	USPAT; US-PGPUB	2002/01/31 16:23
4	L4	8572	state$ adj1 information$	USPAT; US-PGPUB	2002/01/31 16:25
5	L5	1730	1 same 2	USPAT; US-PGPUB	2002/01/31 14:30
6	L6	233	5 same 3	USPAT; US-PGPUB	2002/01/31 14:30
7	L7	0	6 same 4	USPAT; US-PGPUB	2002/01/31 14:31
8	L8	34	6 and 4	USPAT; US-PGPUB	2002/01/31 14:32
9	L9	17643	data adj1 file$	USPAT; US-PGPUB	2002/01/31 14:35
10	L11	585	detect$4 with (state$ adj differen$4)	USPAT; US-PGPUB	2002/01/31 14:44
11	L12	2447	fail$4 adj4 computer$	USPAT; US-PGPUB	2002/01/31 14:46
12	L13	1	10 and 11	USPAT; US-PGPUB	2002/01/31 14:50
13	L14	4	10 and 12	USPAT; US-PGPUB	2002/01/31 15:02
14	L15	521	2 adj3 program$	USPAT; US-PGPUB	2002/01/31 15:05
15	L16	11	hard adj disk$ adj configurat$4	USPAT; US-PGPUB	2002/01/31 15:08
16	L17	449	3 adj3 computer$	USPAT; US-PGPUB	2002/01/31 15:10
17	L18	1	10 and 15	USPAT; US-PGPUB	2002/01/31 15:11
18	L19	0	10 and 16	USPAT; US-PGPUB	2002/01/31 15:11
19	L20	2	10 and 17	USPAT; US-PGPUB	2002/01/31 15:16

	L #	Hits	Search Text	DBs	Time Stamp
20	L21	604	2 adj2 computer$	USPAT; US-PGPUB	2002/01/31 15:17
21	L22	25	21 with 1	USPAT; US-PGPUB	2002/01/31 15:18
22	L23	1	22 and framework	USPAT; US-PGPUB	2002/01/31 15:18
23	L10	15	8 and 9	USPAT; US-PGPUB	2002/01/31 15:31
24	L24	19	8 not 10	USPAT; US-PGPUB	2002/01/31 15:43
25	L25	2499	(714/?).ccls.	USPAT; US-PGPUB	2002/01/31 15:43
26	L26	50	6 and 25	USPAT; US-PGPUB	2002/01/31 15:43
27	L27	47	26 not 8	USPAT; US-PGPUB	2002/01/31 15:43
28	L28	293269	recover$4	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:22
29	L29	36881	backup or back-up or (backing adj1 up)	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:22
30	L30	80194	restor$4	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:23
31	L31	6115	state$ adj1 information$	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:25
32	L32	955	28 same 29	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:25
33	L33	110	32 same 30	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:26
34	L34	3	33 same 31	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:26

DOCUMENT-IDENTIFIER: US 6158019 A
TITLE: System and apparatus for merging a write event journal and an original storage to produce an updated storage using an event map

---------- KWIC ---------

ABPL:
A method and apparatus for **restoring an updated computer** storage from a journal of write events and a copy of an original storage generates an event map from the journal of write events. The event map permits efficient combination of the contents of the write event journal and the original storage. The event map also enables translation of the event journal into a delta expressing the differences between the original and updated storages. The event map similarly permits efficient merging of a write event journal and an original file stored streaming tape.

BSPR:
A write event comprises, for example, a storage indicator indicating what storage component or device the write applies to, a position indicator within the component or event as an offset telling where in the storage the write occurred, and the data (e.g., event data) which was written. Various embodiments also include time or sequence markers for synchronization to identify or select points in time and to coordinate **state information** across storage boundaries. A collection of write events is known as an event log, referred to as an event journal when the event log is stored on a storage device.

BSPR:
File mirroring systems are another type of mirroring. It is known in the art to have a many-to-one mirror, for example when a single system provides a logical mirroring service for a number of network servers. The single system collects change from the servers on a network. The change is stored or applied to a hard disk cache of active **data files**. Periodically the **data file** versions

are backed up to tape. File mirror systems, however, share the same deficiencies as mirror systems because all active data must be stored on disk and included archive service provides random archive granularity. Also as indicated above, mirroring systems maintain a duplicate copy of the storage in case the primary storage system fails.

BSPR:
U.S. Pat. No. 5,086,502 to Malcolm describes an apparatus and method for recording an event journal to a **backup** storage means to provide **backup** to a primary storage drive. The Malcolm patent, however, explicitly requires that a random access storage device hold the base file. It is important to note that this process occurs in the order of the events recorded in the event journal. These events are recorded in the order that changes were made to the base file, and therefore are random with respect to position in the base stream. If the Malcolm system fails, a sequential list of write operations is replayed on a copy of the original data to **restore** the data set to the latest working instance. It is well known in the art to store write events and combine them with a **backup** copy to **recover** a database as of the latest instant. Thus, this technique requires first installing the base copy on random access media and second repeating all write events to the base copy on the disk. Accordingly, data **recovery** using Malcolm journals is restricted to randomly seekable and writeable storage means.

BSPR:
It is also well known in the art to use streaming media to backup **data files**. For example, prior art systems operate by copying **data files** to a streaming media. Streaming media is preferred primarily because of its low cost. Backup systems tend to be used for infrequent retrieval, and when such retrieval is required, data is usually required in the order in which it was recorded.

DEPR:
The event map according to the present invention is useful for a variety of purposes. It is well known to create a backup of a computer by copying an

original storage to a streaming media. Prior art systems describe methods for recovering from an event journal by copying a backup onto a hard disk and "replaying" the events in an event journal. This technique only works, however, if the number of events in the event journal is small enough to replay in a reasonable amount of time. For example, if a large volume of changes are stored in an event journal, replaying the entire event journal to recreate a file could take a prohibitively long time. Thus, such a technique is impractical for sustained off-site backup maintenance. The requirement to periodically refresh the entire backup creates a huge amount of network traffic and disqualifies this method from use for large systems. In addition, the prior art systems require the intermediate step of placing the original **data file** on a seekable medium prior to replaying the event journal to recreate a file. On conventional tape back-up systems, however, only a small amount of disk space is available, if at all, and thus the original file cannot be placed on disk for merging with an event journal as is done via an event map according to the present invention.

DOCUMENT-IDENTIFIER: US 5764877 A
TITLE: Media recovery with time-split B-trees

---------- KWIC ---------

DEPR:
BACKUP ROOT: the location of the history root of the last complete **backup**. This determines where media **recovery** finds the **backup that should be restored**.

DEPR:
Backup produces stable testable state that allow s it to be resumed across system crashes. The history nodes written a re stable, as are the log records describing the updating of index nodes. The nodes changed since the last backup are all recorded on the media recovery log. Hence, the NCV can be recovered. The BSB redundantly contains recent backup **state information** that makes resumption fast.

DEPR:
First, normal database **recovery** is performed, bringing all nodes up to the state as of the time of failure. Recall that TSB-tree **backup** splits are not undone, and hence all such splits will be redone where necessary. Normal database activity can resume at this point. BSB updates are described via log records. Hence, the BSB is **restored** as a result of database **recovery**.

DEPR:
The NCV must also be **restored**. Two methods are possible. (i) The log is accessed in a separate pass to bring the NCV's **backup** up to date. (ii) NCV restoration can be done during the log scan for normal database **recovery** if NCV updating is made a standard part of redo **recovery**.

DEPR:
If **backup** was in progress when the crash occurred, the present invention needs to **restore** the Sweep Cursor. The crash **recovery** log is searched back from its end. The first **backup** log record encountered indicates the last completed **backup** and its HIGHKEY provides the value for Log Key and for Last Key.

DEPR:
When there is a media failure, the current database is reconstructed from the most recent accessible history nodes and the media **recovery** log. The first step is to **restore** all damaged current nodes that have **backups** from the most recent accessible history nodes. The way in which this proceeds will differ depending on the nature and extent of the failure and is discussed in the next subsections. After the **restore** step is completed, and prior to applying the log, failed nodes that existed at the time of the last **backup all have been restored** with valid past states.

DEPR:
With independent redo for current nodes, each of the **restored** nodes can be rolled forward based solely on their log records and their **restored** state. The log will contain sufficient information to regenerate all nodes that do not have **backup** copies in the history database. These were created only via key splitting. Their initial contents have been stored in the log and they can be re-created without access to **backup** versions. Applying the media **recovery** log proceeds exactly like ordinary redo **recovery** after a system crash. The only difference is that the media failure redo scan starts at the media safe point, stored in the BSB, as opposed to the crash **recovery** safe point.

DEPR:
A **restored** database, or substantial part thereof, needs to be relocated to new stable (disk) storage during the restoration step. When the **recovery** log is applied to the **backup** database, we translate the old locations of current nodes, as recorded in the log, to the relocated locations of the **restored** **backup** versions, and apply the log records to the relocated nodes.

DEPR:
If the key range of data in the corrupted node is known, the present invention can readily find a **backup** version in the history database. The present invention uses the key range information to search for the node in the TSB-tree and then searches back in time for the history node. The present

invention
uses this history node to **restore** the corrupted current node. This
node is
then rolled forward by applying the **recovery** log.

DOCUMENT-IDENTIFIER: US 5675802 A
TITLE: Version control system for geographically distributed software development

---------- KWIC ---------

BSPR:
In accordance with another aspect of the invention, the data processing system includes an incremental data **recovery** method. The system includes a **backup** storage device for storing a **backup** copy of the local replica. If the local replica is lost due to a hardware or software failure, the local development site may be **recovered by restoring the backup** copy of the local replica to the storage device and by importing missing files, branches, versions and meta-data from the remote replicas. In particular, the local development site incrementally imports missing files, branches, versions and meta-data from the remote replicas as part of the periodic updating performed by the exchanger.

DEPR:
As noted above, each VOB stores user defined meta-data, which includes version labels, attributes and hyperlinks. Version labels are mnemonic names for particular versions. For example, foo.c version 21 might be tagged with the label "RLS2" in order to indicate that the version was used in the build of the second release. Attributes are name/value pairs, which can be attached to individual versions, entire branches, or entire file elements. Attributes are often used to represent **state information** about a version, for purposes of integrating process-control mechanisms with the version control system. For example, an attribute may be attached to a source file version in order to indicate the bugs fixed by that version. Hyperlinks enable users to define structured or ad hoc relationships between pairs of objects. For example, foo.c can point to foo.doc via a "design.sub.-- for" hyperlink. Hyperlinks are useful for requirements tracking and, with a graphic display of the relationship network, navigating between related objects.

DEPR:
The data processing system of the invention also includes a **backup recovery** mechanism that combines local **backup** restoration for retrieving the bulk of data lost in a failure with incremental restoration from the remote replicas. When a replica is destroyed, it is initially **restored** from a copy of the replica stored in a local **backup** storage device. Any changes logged by remote replicas, but not **restored** by the local **backup,** are then sent from the remote replica that logged them and replayed at the failed replica. The incremental changes are sent as part of the periodic updating process between replicas. As a result, the incremental restoration phase is handled completely automatically and no special communication between the replicas is required to perform incremental restoration. The remote replicas therefore act as incremental **backups** of the failed replica. This mechanism **restores** the maximum number of changes to a failed replica, while minimizing the amount of data that needs to be retrieved from remote replicas.

DEPR:
When a replica is **restored from a backup** storage device, it is not possible to immediately permit users to access the **restored** replica without risking the consistency of the VOB family. Remote replicas in the VOB family may have imported operations that originated at the **restored** replica before its failure, but that were made after the **backup (and not therefore recovered from the backup** copy). Any new operations originated at the **restored** replica risk reusing the same virtual timestamps as those used by other operations already imported by remote replicas, thereby defeating the updating process. The incremental changes must therefore be **restored** to prevent such occurrences.

DEPR:
FIG. 9 is an example of an incremental **recovery** process implemented by each exchanger in geographically distributed data processing systems when one site loses its replica to a hardware or software failure and the **backup** copy

has been **restored** via the **backup** device. The **recovery** process requires the **restored** replica A (124) to re-import any operations (132) it originated and that are possessed by remote replicas B, C, D (126, 128, 130) before permitting users to perform new operations on the **restored** replica. The **recovery** process piggybacks on the normal updating process performed by each exchanger.

DEPR:
A real timestamp (clock time) is also added to the special log entry created by the **restored** replica to handle the situation where the **restored** replica is re-**restored from a backup** device before completing its **recovery**. It is only necessary for the remote replicas to acknowledge the latest **recovery** phase of the **restored** replica. The remote replicas therefore only reset their virtual timestamp tables in response to a special log entry with a real timestamp later than any previously seen.

DOCUMENT-IDENTIFIER: US 5317752 A
TITLE: Fault-tolerant computer system with auto-restart after power-fall

---------- KWIC ---------

DEPR:
The powerfail/autorestart procedure allows all applications to save state that may be resumed when power is **restored**. As noted above, the procedure requires the battery **back-up** 162, 163 to provide system power for a length of time needed to execute an orderly shutdown with the saving of state. By default, applications are not cognizant of the loss of power to the system. In order to allow for the saving of essential state, and later resumption on power restoration, the application must be configured to receive notification from the powerfail/autorestart procedure. Applications so configured may enhance the level of transparency to power loss, and **recover** from the time-latency intransparency already discussed.

DEPR:
The next call is done with a code of PFDUMP, and an argument which is the address of the save area reserved as described in the previous paragraph. The device is responsible of recopying its **state information** into the save area. The powerfail/autorestart procedure will then insure the safety of this data.

DEPR:
Finally, when a save area was requested by the driver for the device, a fourth call will be made to powerfail with a command parameter of PFDUMP and an argument parameter of the address of the save area in which to dump the device state (as requested by the PFQUIESCE call). The device driver is then responsible for copying its **state information** into the save area before returning from this call. The format of the information in the save area is device specific and not defined by the kernel, other than overhead information in the save area is device specific and not defined by the kernel, other than overhead information kept by the kernel to identify this area. Drivers for

devices which do not contain powerfail partitions aliased by PFDEV should power down their device following completion of the PFDUMP call.

DEPR:
Certain device drivers may want to make special cases of the two types of shutdown procedure that can occur, "resume on restart" or "reboot on restart". Device dumps are not actually written to disk if the restart type is "reboot on restart", as this **state information** will not be necessary to the reboot. However, the device driver is not cognizant of this fact. That is, the interaction between restart procedure and the device driver is identical for both "reboot on restart" and "resume on restart". A device driver may determine the type of shutdown in progress by examining a data structure which is accessible in the device driver's address space. The values of the flags indicating the current settings of the kernel powerfail/autorestart procedure switches, as well as the numeric parameters, are included in a file "sys/rolex/pwrfail.h".

DEPR:
First, the I/O processor state is restored. This state includes the access validation RAM information that represents a portion of the device state. Then the device identification routine is called for each device. Then, the powerfail routine of the driver for each device will be called with a command of PFINIT. This call is intended for the disk driver for disk 148 only. All other drivers may ignore it. At this point, the disk driver initializes itself so the device state for the other device drivers may be read in off the powerfail dump device, PFDEV. Next, the powerfail routine of the driver for each device will be called with a command of PFRESTORE and an argument of the address of the save area requested by the device during shutdown. If no save area was requested for this device by the driver, the argument will be NULL. This call to the powerfail routine is to reload any **state information**.

DEPR:
Upon restart, if the "reboot on restart" option is chosen, the disk system 148 will be involved in bringing a new copy of the operating system off from the disk. Rather than pursuing a normal boot when AC power is restored, the

essential system image contained on PFDEV is loaded into memory 14, 15 or 16.
This essential system image contains device **state information** which may be passed to a given device, as previously mentioned, and then discarded. The other portion of the essential system state consists of an actual core image, possibly in noncontiguous portions. If an I/O error occurs when restoring the essential system state, control is passed back to the initial boot sequence and a fresh copy of the operating system is loaded, regardless of the value of the kernel parameter which requests "resume on restart".

CLPR:
7. A method according to claim 6 wherein said computer system includes a plurality of devices external to said CPU and a nonvolatile memory, said CPU executes processes during normal operation and wherein said shutdown process includes writing **state information** to nonvolatile memory, wherein said **state information** includes the state of the processes executing on said computer system and device **state information**.

CLPR:
10. A method according to claim 6 wherein said shutdown process includes the steps of copying **state information** from said CPU, processes and devices of said system to selected locations in volatile memory and then writing said selected locations to non-volatile memory.

CLPR:
12. A method according to claim 11 wherein said shutdown process includes the steps of copying **state information** from said CPU, processes and devices of said system to selected locations in volatile memory and then writing said selected locations to disk.

CLPV:
(b) detecting a failure of said main power supply and, in response thereto, providing power to said computer system from the backup power supply and executing a shutdown procedure in said CPU, said shutdown procedure including first warning said processes of an impending shutdown of the computer system,

said processes responding to said warning in a manner varying from process to
process, and then copying **state information** of said computer system from said
memory to said non-volatile storage, wherein said **state information** includes
state information of the processes and **state information** of the devices;

PAT-NO: JP411212845A

DOCUMENT-IDENTIFIER: JP 11212845 A

TITLE: DEVICE AND METHOD FOR BACKUP DATA MANAGEMENT AND RECORDING MEDIUM

PUBN-DATE: August 6, 1999

INVENTOR-INFORMATION:

NAME	COUNTRY
KOSUGI, HAJIME	N/A

ASSIGNEE-INFORMATION:

NAME	COUNTRY
NEC ENG LTD	N/A

APPL-NO: JP10009666

APPL-DATE: January 21, 1998

INT-CL (IPC): G06F012/00

ABSTRACT:

PROBLEM TO BE SOLVED: To make it possible to shorten a time needed for data restoration.

SOLUTION: A processor 10 uses not only a magnetic tape device 12 which is lower in speed than accessing to a hard disk device 13 and has a large capacity but also the hard disk device 13 and retains data 15 stored in the hard disk device 13 as backup data 16 when performing backup of the data 15. Also, the processor 10 registers data restoration state information for indicating whether or not a data backed up party is the magnetic tape device 12 or both with a management table 17 for every backed up data. The processor 10 shortens a time needed for **recovery** by giving a priority to the **hard disk** device 13 and performing **recovery** when the data of a backup object are backed up in the **hard disk** device 13 based on the data storage **state information** registered in the management table 17.